Solución:
Examen de Introducción a las Redes de Computadoras y Comunicación de Datos
(ref: sirc0608.doc)
5 de agosto de 2006

Preguntas Teóricas

Pregunta 1 (5 puntos)

Enuncie los resultados de Nyquist y Shannon vistos en el teórico. Comente sobre las hipótesis de cada uno de ellos.

Respuesta:

Teorema de Nyquist
Hipótesis:

- Canal de comunicaciones de ancho de banda limitado H, sin ruido de ningún tipo
- La señal transmitida por el canal es una señal digital de V niveles discretos

Resultado:
La máxima tasa de transmisión de bits alcanzable es → $M = 2 H \log_2 (V)$

Ley de Shannon
Hipótesis:

- Canal de ancho de banda H
- Canal sometido a ruido gaussiano de potencia N
- Señal de transmisión de potencia S

Resultado:
La capacidad del canal (en bits), tiene un máximo teórico → $C = 2 H \log2 [1 + S/N]$

Pregunta 2 (10 puntos)

Explique por qué es necesario el protocolo ARP y describa cómo funciona.

Respuesta:

ARP son las siglas en inglés de Address Resolution Protocol (Protocolo de resolución de direcciones).

Es un protocolo de nivel de red responsable de encontrar la dirección hardware (Ethernet MAC) que corresponde a una determinada dirección IP. Para ello se envía un paquete (ARP request) a la dirección de multidifusión de la red (broadcast (MAC = ff ff ff ff ff ff)) conteniendo la dirección IP por la que se pregunta, y se espera a que esa máquina (u otra) responda (ARP reply) con la dirección Ethernet que le corresponde.

Cada máquina mantiene un caché con las direcciones traducidas para reducir el retardo y la carga. ARP permite a la dirección de Internet ser independiente de la dirección Ethernet, pero esto solo funciona si todas las máquinas lo soportan.

En Ethernet, la capa de enlace trabaja con direcciones físicas. El protocolo ARP se encarga de traducir las direcciones IP a direcciones MAC (direcciones físicas).Para realizar ésta conversión, el nivel de enlace utiliza las tablas ARP, cada interfaz tiene tanto una dirección IP como una dirección física MAC.

Pregunta 3 (10 puntos)

Describa detalladamente el funcionamiento de un protocolo de enrutamiento de tipo DISTANCE-VECTOR (VECTOR-DISTANCIA). Mencione posibles ventajas y desventajas del mismo.

Respuesta:

Algoritmos de “Distancia-Vector” (distancevector)
Trabajan guardando una tabla con las distancias óptimas para cada destino, y periódicamente las comparten con sus routers vecinos inmediatos.

Ejemplo:

- RIP: el primer protocolo dinámico en Internet

Enrutamiento Distancia-Vector: Algoritmo de Bellman-Ford

Algoritmo distancia-vector, el mas conocido y aplicado
Modelo de tabla de enrutamiento:
Toma la forma de un “vector de distancias” →

To	A
A	0
B	12
C	25
D	40
E	14
F	23
G	18
H	17
I	21
J	9
K	24
L	29

Routers se inician con una tabla de enrutamiento “vacía”

- En realidad, saben llegar a ellos mismos.
- Inicializan un timer “T”, cada vez que el timer se vence, envían a sus vecinos por todos sus enlaces su tabla.
- Escuchan en todos sus enlaces por las tablas publicadas por sus vecinos. Cuando reciben una tabla de un vecino, la comparan con la propia y actualizan la propia.

Actualización de las Tablas:

- Sea T0 la tabla propia y T1 la tabla recibida del router vecino R
- Para cada ruta a un host A en T1 a través de R
- Si no existe en T0 => agrego una ruta hacia A por R, incrementando la distancia adecuadamente.
- Si existe en T0, pero con una métrica mayor => la sustituyo por la ruta por R actualizando la métrica
- Si existe en T0 la misma ruta, la marco como “refrescada”
- Luego de recibidos updates por todas las interfaces, borro todas las rutas “no refrescadas”

Problemas con el Bellman-Ford

- Conteo a Infinito
- Las malas noticias se propagan lento
- Cuando un enlace cae, hay un gran numero de iteraciones hasta que todos los nodos se dan cuenta que hay un problema.

Pregunta 4 (10 puntos)

Para un nuevo proyecto a la empresa X le ha sido asignado el siguiente bloque de direcciones IP: 200.40.224.0/25.

a) ¿Cuántas direcciones contiene este bloque? ¿Cuántas son efectivamente utilizables?
b) Proponga una subdivisión de este bloque sabiendo que se necesitan numerar 3 redes de 18, 11 y 9 hosts y tres enlaces punto a punto. Liste las redes que quedan sin utilizar.

Respuesta:

a) ¿Cuántas direcciones contiene este bloque? ¿Cuántas son efectivamente utilizables?

El bloque 200.40.224.0/25 contiene 7 bits de direccionamiento

- en total 128 direcciones
- direcciones utilizables (quitando la de red 00000 y la broadcast 11111) 126

b) Proponga una subdivisión de este bloque sabiendo que se necesitan numerar 3 redes de 18, 11 y 9 hosts y tres enlaces punto a punto. Liste las redes que quedan sin utilizar.

- 18 hosts, se debe incluir dirección broadcast y de red. Por lo que se requiere soporte más de 20 direcciones. Se requieren 5 bits (implica 32 direcciones)
- 11 hosts, se debe incluir dirección broadcast y de red. Por lo que se requiere soporte más de 13 direcciones. Se requieren 4 bits (implica 16 direcciones)
- 9 hosts se debe incluir dirección broadcast y de red. Por lo que se requiere soporte más de 20 direcciones. Se requieren 4 bits (implica 16 direcciones)
- tres enlaces punto a punto. Se debe incluir dirección broadcast y de red. Por lo que se requiere soporte más de 4 direcciones cada uno. Se requieren 2 bits (implica 4 direcciones)

Numeración de las redes:

Cantidad hosts	**Cantidad IP's**	**Cantidad bits**	**IP asignadas**
18	20	5	200.40.224.0/27
11	13	4	200.40.224.32/28
9	11	4	200.40.224.48/28
2	4	2	200.40.224.64/30
2	4	2	200.40.224.68/30
2	4	2	200.40.224.72/30
Sin utilizar			200.40.224.76/30
Sin utilizar			200.40.224.80/29
Sin utilizar			200.40.224.88/29
Sin utilizar			200.40.224.96/29
Sin utilizar			200.40.224.104/29
Sin utilizar			200.40.224.112/29
Sin utilizar			200.40.224.120/29

Pregunta 5 (5 puntos)

Describa los conceptos fundamentales de MPLS: la etiqueta, el LSP y los diferentes roles que cumplen los routers en el modelo MPLS.

Respuesta:

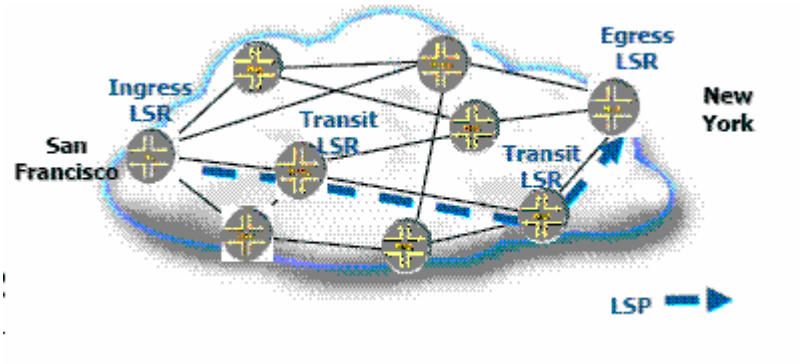
Etiqueta:

Label	Exp. bits	BS	TTL
20	3	1	8

- Label:
  - Identificador de flujo, índice en tabla
- Experimental bits (QoS)
  - Clase de servicio
- BS
  - Stacking de etiquetas
- TTL
  - Usado de la misma forma que en IP

Label Switched Path (LSP)

- Túnel unidireccional (simplex) a través de la red, formado por saltos controlados por las etiquetas



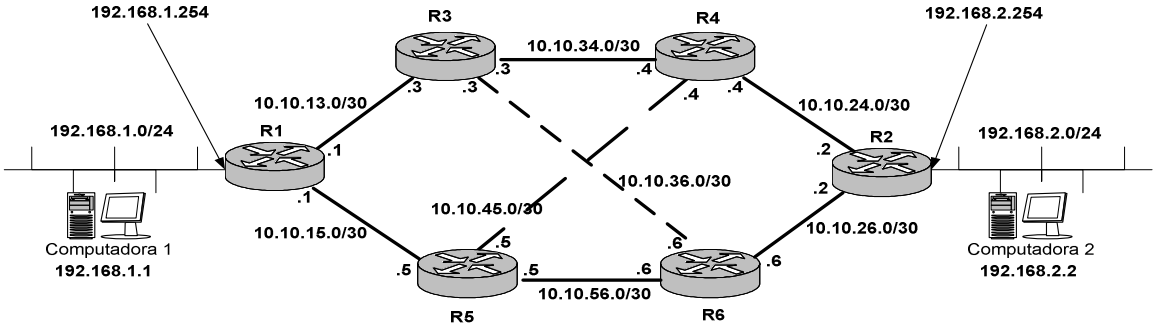
Label Switch Router

- **Ingress LSR** (“head-end LSR”)
  - Examina los paquetes IP entrantes y los clasifica asignándolos a una FEC
  - Genera el encabezado MPLS y asigna la etiqueta inicial
- **Transit LSR**
  - Reenvía los paquetes MPLS usando la etiqueta y haciendo swapping
- **Egress LSR** (“tail-end LSR”)
  - Remueve el encabezado MPLS.

Problemas Prácticos

Problema 1 (30 puntos)

En la figura se muestra la topología que interconecta dos redes LAN. La red de interconexión está formada por 6 *routers* (R1 .. R6) unidos por enlaces punto a punto según se muestra. Se detalla para cada enlace su plan de numeración. Para las redes LAN se detalla también el plan de numeración correspondiente así como las direcciones IP asignadas a dos computadoras presentes en ellas y el *default gateway* en cada caso (192.168.1.254 y 192.168.2.254 respectivamente).



La información relevante de enrutamiento que posee cada *router* es la siguiente:

Router	Destino	A través de	Costo en dicho enlace
R1	192.168.1.0/24	Directamente conectada	-
	192.168.2.0/24	10.10.15.5	1
	192.168.2.0/24	10.10.13.3	2
R2	192.168.1.0/24	10.10.26.6	1
	192.168.1.0/24	10.10.24.4	2
	192.168.2.0/24	Directamente conectada	-
R3	192.168.1.0/24	10.10.13.1	2
	192.168.1.0/24	10.10.36.6	5
	192.168.2.0/24	10.10.34.4	2
	192.168.2.0/24	10.10.36.6	5
R4	192.168.1.0/24	10.10.45.5	2
	192.168.1.0/24	10.10.34.3	6
	192.168.2.0/24	10.10.24.2	1
	192.168.2.0/24	10.10.45.5	3
R5	192.168.1.0/24	10.10.15.1	1
	192.168.1.0/24	10.10.45.4	4
	192.168.2.0/24	10.10.45.4	1
	192.168.2.0/24	10.10.56.6	2
R6	192.168.1.0/24	10.10.56.5	1
	192.168.1.0/24	10.10.36.3	4
	192.168.2.0/24	10.10.26.2	2
	192.168.2.0/24	10.10.36.3	4

a) Explique la utilidad del campo *Time To Live* del encabezado IP.

b) Se ejecuta en la Computadora 2, el siguiente comando:

traceroute 192.168.1.1

Considere que:
el comando que se ejecuta está completamente basado en mensajes ICMP.
toda la red funciona sin inconvenientes y el enrutamiento se encuentra estable.
todo mensaje ICMP generado por cualquier *router* es enviado en un paquete IP cuya dirección origen es la perteneciente al enlace punto a punto por donde ingresó al *router* el paquete que lo motivó.

Indique qué observa en su *terminal* la persona que ejecuta dicho comando.
Nota: No es necesario (ni posible) especificar nada respecto a tiempos involucrados.

c) Detalle claramente cada uno de los tipos de mensajes ICMP que se generan en la red como consecuencia de ejecutarse el comando de la parte **b)**.

d) Si ocurre la siguiente secuencia de eventos:
   1. Toda la red opera correctamente
   2. El *router* R4 deja de funcionar
   3. El enrutamiento en toda la red converge
   4. Se ejecuta el comando en la Computadora 2 nuevamente el comando:

traceroute 192.168.1.1

Indique cómo cambia su respuesta de la parte **b)**. Justifique su respuesta.

e) Si ocurre la siguiente secuencia de eventos:
   1. Toda la red opera correctamente
   2. El *router* R5 deja de funcionar
   3. El enrutamiento en toda la red converge
   4. Se ejecuta el comando en la Computadora 2 nuevamente el comando:

traceroute 192.168.1.1

Indique cómo cambia su respuesta de la parte **b)**. Justifique su respuesta.

Solución

a)

Permite que los paquetes no tengan vida “infinita” en la red. Cada vez que un paquete arriba a un nodo, el valor del campo TTL se decrementa en 1.
Si el valor obtenido es distinto de 0, se procesa como es habitual (según dicho nodo sea el destinatario de dicho paquete o no).
Si es igual a 0 y dicho nodo no es el destinatario de dicho paquete, el mismo se descarta y se informa de dicha situación al origen del paquete. Para informar esto, se envía un tipo particular de mensajes ICMP (Internet Control Message Protocol), conocido como “*Time Exceeded*” (tipo 11). Este mensaje se envía como payload de un paquete IP que tiene como dirección IP origen quien lo genera y dirección IP destino, la de quién generó el paquete IP al que le llegó a 0 su TTL.
De esta forma, el campo TTL permite “limpiar” las redes de paquetes que, por problemas de enrutamiento por ejemplo, no logran alcanzar su destino.
Por otro lado, forzando a valores particulares el campo TTL de los paquetes podemos hacer que ciertos nodos indiquen que esos paquetes han llegado a ellos. Esto es utilizado por herramientas como el “traceroute” para conocer el camino que recorren los paquetes, en particular, el camino de ida entre quien lo ejecuta y el destino. No permite afirmar nada respecto al camino de vuelta, pues el enrutamiento puede ser tal que no coincida con el de ida.

b)
Se asume que la ejecución de dicho comando implicará en cada paso el envío de 3 paquetes.
De acuerdo a la información de enrutamiento brindada, el camino de menor costo para ir de la red 192.168.2.0/24 a la red 192.168.1.0/24 es R2 → R6 → R5 → R1.
La salida será entonces:

 # traceroute 192.168.1.1
1 t1 t2 t3 192.168.2.254 "+"
2 t4 t5 t6 10.10.26.6 "++"
3 t7 t8 t9 10.10.56.5
4 t10 t11 t12 10.10.15.1
5 t13 t14 t15 192.168.1.1 "+++"
Trace complete.
#

Los números *t1 .. t15* están involucrados con retardos.

c)
La Computadora 2 genera siempre mensajes ICMP tipo 8, *Echo*, que viajan en paquetes IP con dirección IP origen 192.168.2.2 y dirección IP destino 192.168.1.1.
Los tres primeros llevan el campo TTL = 1. Al llegar a R2, se decrementa el TTL en 1 (pasa a valer 0) y R2 no es el destino de dichos paquetes, por lo que los paquetes se descartan y se genera tres mensajes ICMP tipo 11, *Time Exceeded*, los que viajan dentro de tres paquetes IP con dirección IP origen 192.168.2.1 y dirección IP destino, 192.168.2.2. Estos paquetes hacen que se despliegue la primer línea (identificada como "+").
Luego de esto, la Computadora 2 genera tres nuevos paquetes similares a los primeros, pero con TTL = 2. Ello ocasiona que ahora R6 genere los tres mensajes ICMP tipo 11, ocasionando la línea identificada como "++".
El comportamiento es similar para R5 y R1, con TTL = 3 y TTL = 4 respectivamente.
Los paquetes con TTL = 5 sí llegan a la Computadora 1 y por lo tanto, ésta genera 3 mensajes ICMP tipo 0, Echo Reply, a partir de los cuales se genera la línea "+++"

d)
La respuesta es la misma, pues R4 no es utilizado en el camino de ida.

e)
El camino hasta R1 (y hasta la Computadora 1) es a través de R3

 # traceroute 192.168.1.1
1 t1' t2' t3' 192.168.2.254
2 t4' t5' t6' 10.10.26.6
3 t7' t8' t9' 10.10.36.3
4 t10' t11' t12' 10.10.13.1
5 t13' t14' t15' 192.168.1.1
Trace complete.
#

Los números *t1' .. t15'* están involucrados con retardos.

Problema 2 (30 puntos)

Se desea programar el juego de *La batalla Naval* (por turnos) en una red TCP/IP, a través de un protocolo no confiable UDP. La idea es que los jugadores se encuentran en diferentes computadores, e interactúan realizando sus jugadas a través de la red.

Se cuenta con las siguientes funciones que responden a invocaciones a procedimientos en el computador local y no requiere el uso de señales con el entorno:

- *InicializarJuego()* procedimiento que genera un tablero de 8x8 donde se encuentran ubicados 1 barco de 4 casillas, 2 barcos de 3 casillas, y 3 barcos de 2 casillas, y genera un tablero de 8x8 vacío donde se representan las jugadas realizadas por el jugador local, lo despliega en la pantalla del computador.
- *AplicarJugada(x,y): resultado* donde resultado es un carácter "A", "T", "H" o "P" que significa Agua, Tocado, Hundido, Perdi, respectivamente e indica en el tablero que contiene los barcos seleccionados por el jugador.
- *PedirCoordenada():(int x, int y)* que devuelve dos números entre 1 y 8 ingresado por el jugador, y marca en el tablero que despliega las jugadas efectuadas .
- *DesplegarMensaje(<texto_Mensaje>)* que despliega el texto en la terminal del computador.

Se dispone de un puerto UDP *puertoBN* en el cual se pondrá a funcionar la aplicación del juego en ambos computadores. Debe considerarse que el protocolo de red utilizado no es confiable, por lo que puede ocurrir pérdida o duplicación de los mensajes enviados.

Debe considerarse:

- Inicialmente los jugadores intercambian un mensaje indicando el inicio del juego, y no se aceptará otro mensaje de inicio, hasta que el juego finalice o se baje la aplicación.
- El primer atacante es el que inicia el juego.
- Cuando un computador determine que él otro equipo ganó, enviará un mensaje finalizando el juego.
- Asuma que los dos computadores no inician juego a la vez.

Se pide:

Indique el formato de los mensajes que deben intercambiar los jugadores, indicando qué implica cada uno, considerando además que el juego se desarrolla únicamente entre dos computadores, e implemente en un lenguaje de alto nivel, el juego mencionado.

Nota: se supone conocido el juego "La Batalla Naval" planteado en el problema.

Solución

Se plantea resolver sobre UDP un protocolo que permita mantener las sesiones del juego, permitiendo además detectar la pérdida y duplicación de segmentos.

Se propone utilizar un mecanismo de ARQ Stop&Wait, basado en su simplicidad, y que los requerimientos expuestos no exigen más.

Para la implementación del mismo se propone el siguiente formato de paquete, con los siguientes tipos de paquete:

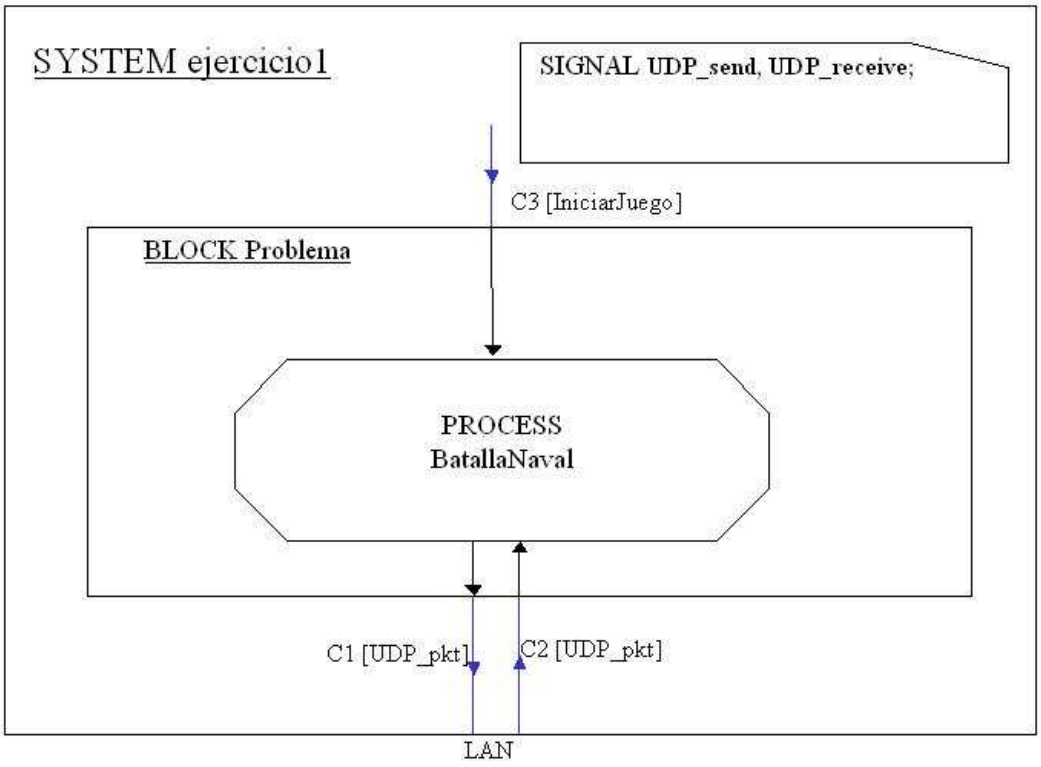
- Para inicio de la partida se utiliza el tipo de paquete `iniSol y iniResp,`
- Para el juego se utiliza los tipos de paquete `BNjug y BNnack,`
- Para finalizar se utiliza el tipo de paquete `BNFin`

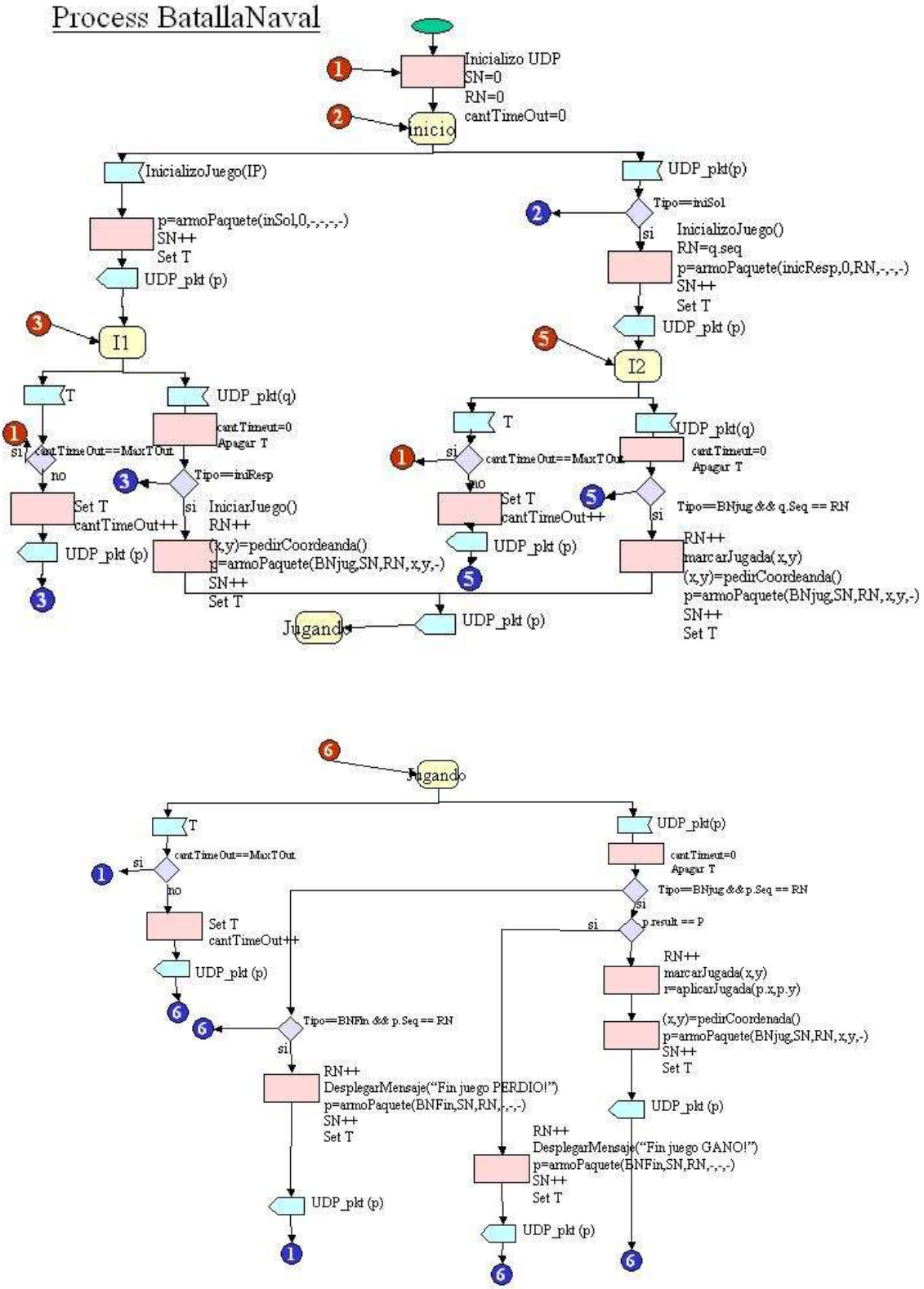
```
enum tipoPaquete {iniSol, iniResp, BNjug, BNnack, BNFin};

typedef struct
{
tipoPaquete tipo;
int seq;
int ack;
char resultado;
int x;
int y;

} UDP_pkt;
```

El tipo de paquete iniSol contiene el número inicial de secuencia en seq.
El tipo de paquete iniResp contiene el número inicial de secuencia en seq y ack del inicio recibido.
El tipo de paquete BNjug contiene seq, ack resultado (que puede ser “A”, “T”, “H”, “P”); y x, y de la próxima jugada.
El tipo de paquete BNnak contiene en ack el paquete que esta faltando en la comunicación.
El tipo de paquete BNFin contiene el número de secuencia seq y ack.




Process BatallaNaval
Inicializo UDP
SN=0
RN=0
cantTimeOut=0
inicio
InicializoJuego(IP)
p=armoPaquete(inSol,0,-,-,-,-)
SN++
Set T
UDP_pkt (p)
I1
T
UDP_pkt(q)
cantTimeout=0
Apagar T
cantTimeOut==MaxTOut
si
no
Tipo==inuResp
Set T
cantTimeOut++
UDP_pkt (p)
IniciarJuego()
RN++
(x,y)=pedirCoordeanda()
p=armoPaquete(BNjug,SN,RN,x,y,-)
SN++
Set T
UDP_pkt(p)
Tipo==iniSol
InicializoJuego()
RN=q.seq
p=armoPaquete(inicResp,0,RN,-,-,-)
SN++
Set T
UDP_pkt (p)
I2
T
UDP_pkt(q)
cantTimeout=0
Apagar T
cantTimeOut==MaxTOut
si
no
Set T
cantTimeOut++
UDP_pkt (p)
Tipo==BNjug && q.Seq == RN
si
RN++
marcarJugada(x,y)
(x,y)=pedirCoordeanda()
p=armoPaquete(BNjug,SN,RN,x,y,-)
SN++
Set T
Jugando
UDP_pkt (p)
Jugando
T
cantTimeOut==MaxTOut
si
no
Set T
cantTimeOut++
UDP_pkt (p)
Tipo==BNFin && p.Seq == RN
si
RN++
DesplegarMensaje("Fin juego PERDIO!")
p=armoPaquete(BNFin,SN,RN,-,-,-)
SN++
Set T
UDP_pkt (p)
UDP_pkt(p)
cantTimeout=0
Apagar T
Tipo==BNjug && p.Seq == RN
si
p.result == P
si
RN++
marcarJugada(x,y)
r=aplicarJugada(p.x,p.y)
(x,y)=pedirCoordenada()
p=armoPaquete(BNjug,SN,RN,x,y,-)
SN++
Set T
UDP_pkt (p)
RN++
DesplegarMensaje("Fin juego GANO!")
p=armoPaquete(BNFin,SN,RN,-,-,-)
SN++
Set T
UDP_pkt (p)